PROCESSADORES PROGRAMÁVEIS PARA 32 INSTRUÇÕES

O projeto de 32 instruções é uma extensão máxima do projeto do processador programável de 03 instruções. A arquitetura básica do processador de 03 instruções é modificada para que programas que contenham blocos de decisões fossem introduzidas e com isso a implementação de algorítmos mais complexos. A introdução do bloco "detetor de zero" vai permitir a resposta da ULA com operações cujo resultado produz um valor zero na saída, bem como o bloco comparador de amplitude que gera 03 saídas "maior, igual e menor" e um segundo bloco comparador o qual produz 02 saídas maior ou igual e menor ou igual. Todos esses indicadores servirão para produzirem um "status" da operação realizada na ULA ou nos testes de comparações. A tabela a seguir define todos os indicadores "flags" e a sua lógica ativa.

Indicador	Flag	Comentário
Zero	Z	Z = "1" qdo a operação realizada na ULA produz saída igual a zero.
Maior	M	M = "1" qdo a operação de comparação realizada entre 2 operandos do registrador de arquivos produzir a saída maior a "1".
Menor	N	N = "1" qdo a operação de comparação realizada entre 2 operandos do registrador de arquivos produzir a saída menor a "1".
Igual	I	I = "1" qdo a operação de comparação realizada entre 2 operandos do registrador de arquivos produzir a saída igual a "1".
Maior ou igual	MEQ	MEQ = "1" qdo a operação de comparação realizada entre 2 operandos do registrador de arquivos produzir a saída maior ou igual a "1".
Menor ou igual	NEQ	NEQ = "1" qdo a operação de comparação realizada entre 2 operandos do registrador de arquivos produzir a saída menor ou igual a "1".
Bit_DO	B0	B0 = "1" qdo o bit LSB do deslocador é igual a "1".
Bit_D7	B7	B7 = "1" qdo o bit MSB do deslocador é igual a "1".
Carry	C	C = "1" qdo a operação realizada na ULA estoura a sua capacidade máxima.
Overflow	OV	OV = "1" qdo a operação entre 2 operandos assinalados realizada na ULA produz um transbordamento na ULA.

Os indicadores informam a unidade de controle sobre o "status" da operação na execução da instrução. O programa feito pelo usuário deve utilizar essa informação para a tomada de decisão. A instrução de Salto foi expandida para 12 novas instruções e 4 bits seguidos dos 4 bits fixos do "opcode" servirão para a criação dessas instruções de saltos a seguir.

Expansão	Mnem.	Operação	Comentário
0000	JZ	Salto se zero	Se a operação realizada na ULA produz zero;
0001	JNZ	Salto se não zero	Se a operação realizada na ULA produz um resultado diferente de zero;
0010	-	-	-
0011	JNEQ	Salto se menor ou igual	Salto se a operação de comparação entre 2 números resulta em menor ou igual;
0100	-	-	-
0101	JMEQ	Salto se maior ou igual	Salto se a operação de comparação entre 2 números resulta em maior ou igual;
0110	-	-	-
0111	-	-	-
1000	JB0	Salto de bit_D0	Salto se bit D0 do deslocador é igual a "1";
1001	JNB0	Salto se bit_NDO	Salto se bit D0 do deslocador é igual a "0";
1010	JB7	Salto se bit_D7	Salto se bit D7 do deslocador é igual a "1";

1011	JNB7	Salto se bit_ND7	Salto se bit D7 do deslocador é igual a "0";
1100	JM	Salto se maior	Salto se a operação de comparação entre 2 números resulta em maior;
1101	JI	Salto se igual	Salto se a operação de comparação entre 2 números resulta em igual;
1110	JN	Salto se menor	Salto se a operação de comparação entre 2 números resulta em menor;
1111	JMP	Salto incondicional	Salto incondicionalmente

O conjunto de instruções com 32 instruções é apresentado a seguir, bem como a tabela com os "opcode" e comentários.

Opcode	Mnem	Operação	Comentário
0000	LD(ra),(d)	RF[ra] ← (d)	Carrega o conteúdo endereçado em d para o RF[ra];
0001	ST(d),(ra)	(d) ← RF[ra]	Armazena o conteúdo de RF[ra] no endereço (d);
0010	ADD(ra),(rb),(rc)	RF[ra] = RF[rb] + RF[rc]	Soma os conteúdos dados por RF[rb,rc] e armazena em RF[ra];
0011	MVI(ra) #c	RF[ra] ← w_data	Transfere o conteúdo imediato em RF[ra];
0100	SUB(ra),(Rb),(rc)	RF[ra] = RF[rb] - RF[rc]	Subtrai os conteúdos dados por RF[rb,rc] e armazena em RF[ra];
0101	JX, offset	X = 0000 a 1111	Salta para o endereço dado em offset;
0110	OUT(ra)	Saída ← RF[ra]	Transfere para a saída o conteúdo de RF[ra];
0111	IN(ra)	RF[ra] ← input_ext	Transfere da entrada o conteúdo para RF[ra];
1000	CMP(ra),(rb)	RF[ra] : RF[Rb]	Compara os conteúdos de RF[ra] com RF[rb];
1001	XOR(ra),(Rb),(rc)	RF[ra] = RF[rb] ⊕ RF[rc]	XOR os conteúdos dados por RF[rb,rc] e armazena em RF[ra];
1010	AND(ra),(Rb),(rc)	RF[ra] = RF[rb].RF[rc]	AND os conteúdos dados por RF[rb,rc] e armazena em RF[ra];
1011	OR(ra),(Rb),(rc)	RF[ra] = RF[rb] + RF[rc]	OR os conteúdos dados por RF[rb,rc] e armazena em RF[ra];
1100	INC(ra)	RF[ra] = RF[ra] - 1	Incrementa o conteúdo de RF[ra];
1101	DEC(ra)	RF[ra] = RF[ra] – 1	Decrementa o conteúdo de RF[ra];
1110	SHX	RF[ra] ← bit	Desloca o conteúdo RF[ra] bit;
1111	HLT	(PC) = (PC)	Pára o processamento.

Assim como foi projetado para a instrução SALTO é adotado o mesmo procedimento para as instruções de deslocamentos, as quais podem ser do tipo: deslocamento para esquerda, para a direita, circular a esquerda e circular a direita. A tabela a seguir apresenta a expansão da instrução SHX para 04 instruções a seguir.

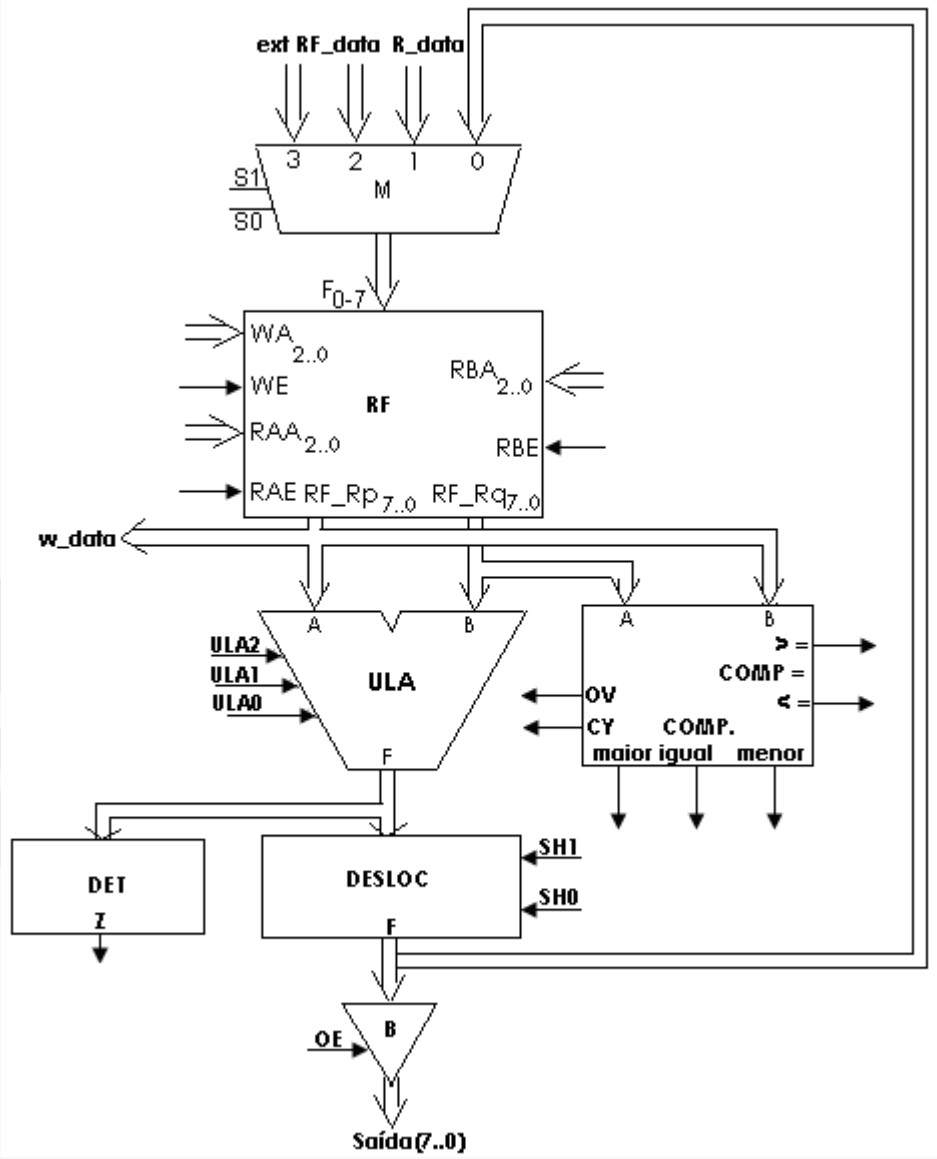
Expansão	Mnem.	Lógica	Comentário
0000	SHL	RF[ra] ← 0	Deslocamento a esquerda do conteúdo do deslocador um bit e introdução do bit D0 igual a zero.
0001	SHR	RF[ra] ← 0	Deslocamento a direita do conteúdo do deslocador um bit e introdução do bit D7 igual a zero.
0010	ROL	RF[ra] ← D7	Deslocamento circular a esquerda do conteúdo do deslocador um bit e introdução do bit D7.
0011	ROR	RF[ra] ← D0	Deslocamento circular a direita do conteúdo do deslocador um bit e introdução do bit D0.

Projeto do fluxo de dados para o processador de 32 instruções.

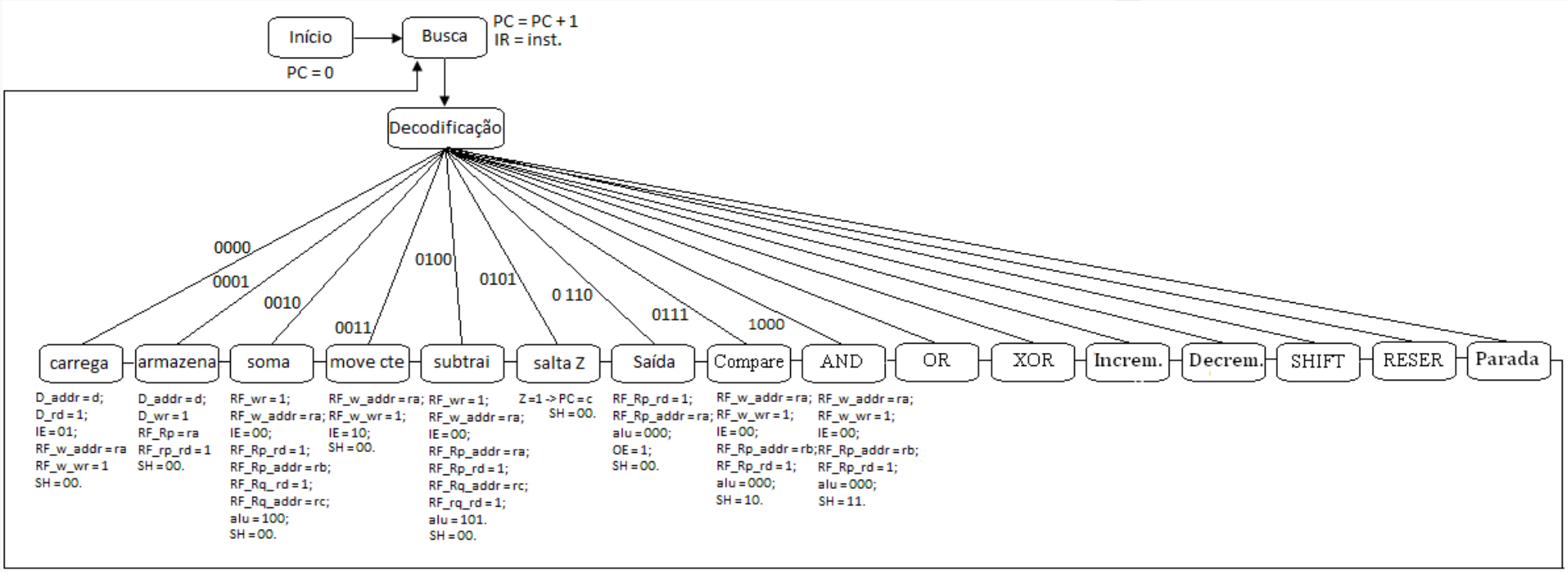
O projeto do fluxo de dados é uma melhoria na arquitetura do processador de 03 instruções apresentado nesse capítulo. Para que fossem expandido o conjunto de instruções, a introdução dos indicadores para a U.C. é fundamental para a tomada de decisão. Os blocos a seguir fazem parte do fluxo de dados para 32 instruções. A saber:

1. Bloco MUX;
2. Bloco RF;
3. Bloco comparador;
4. Bloco detetor de zero;
5. Bloco comparador com igualdade;
6. Bloco ULA;
7. Bloco deslocador;
8. Bloco saída.

Arquitetura do processador de 32 instruções



O diagrama de estados a seguir mostra a busca da instrução, decodificação da instrução e execução.

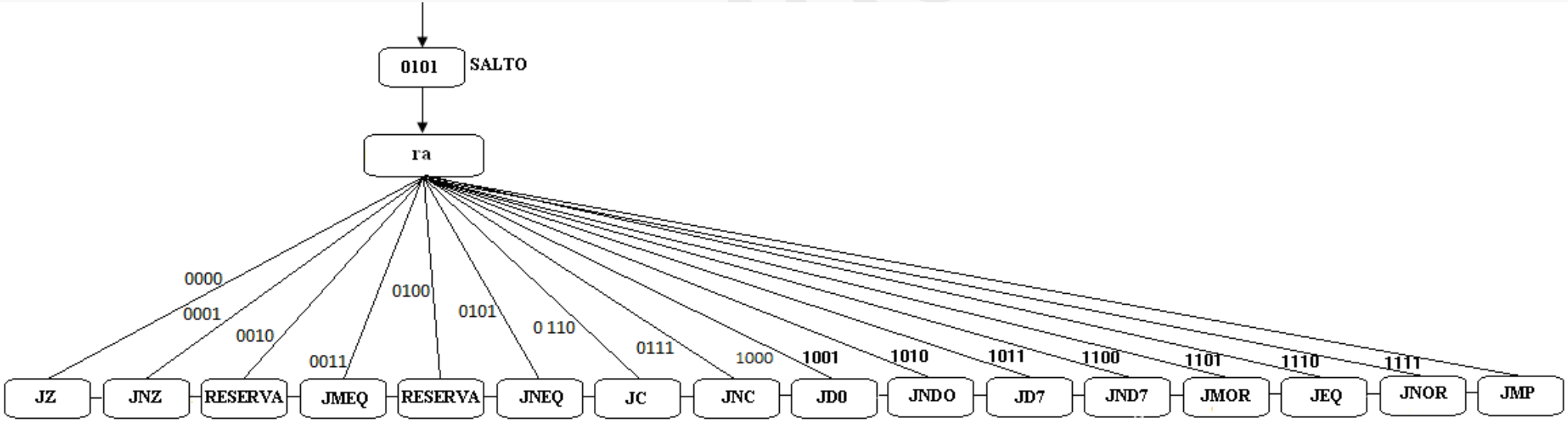


O quadro das palavras de controle geradas pela u.c. para a execução da instrução.

Instrução	Opcode	D_addr	D_rd	D_wr	RF_s	RF_w_addr	RF_w_wr	RF_Rp_rd	RF_Rp_addr	RF_Rq_rd	RF_Rq_addr	ALU	SH	OE
Carga	0	d	1	0	1	ra	1	0	xxxx	0	xxxx	xxx	xx	0
Armazena	1	d	0	1	x	xxxx	0	1	ra	0	xxxx	xxx	xx	0
Soma	2	x..x	0	0	0	ra	1	1	rb	1	rc	100	00	0
Transfer	3	x..x	0	0	2	ra	1	0	xxxx	0	xxxx	Xxx	xx	0
Subtrai	4	x..x	0	0	0	ra	1	1	rb	1	rc	101	00	0
Salto	5	x..x	0	0	0	xxxx	0	0	xxxx	0	xxxx	xxx	xx	x
Saída	6	x..x	0	0	0	xxxx	0	1	ra	0	xxxx	000	00	1
Entrada	7	x..x	0	0	3	ra	1	0	xxxx	0	xxxx	xxx	xx	0
Compare	8	x..x	0	0	0	xxxx	0	1	ra	1	Rb	xxx	xx	0

Instrução	Opcode	D_addr	D_rd	D_wr	RF_s	RF_w_addr	RF_w_wr	RF_Rp_rd	RF_Rp_addr	RF_Rq_rd	RF_Rq_addr	ALU	SH	OE
XOR	9	0	0	0	0	ra	1	1	rb	1	rc	011	00	0
AND	A	0	0	0	0	ra	1	1	rb	1	rc	001	00	0
OR	B	0	0	0	0	ra	1	1	rb	1	rc	010	00	0
INC	C	0	0	0	0	ra	1	1	ra	0	xxxx	111	00	0
DEC	D	0	0	0	0	ra	1	1	ra	0	xxxx	110	00	0
SHIFT	E	-	-	-	-	-	-	-	-	-	-	-	-	-
STOP	F	0	0	0	0	xxxx	0	0	xxxx	0	xxxx	xxx	xx	x

As instruções "salto" e "shift" vão ser tratadas separadamente, pois fazem parte da expansão das instruções. A instrução salto se expande em 12 instruções a saber. O diagrama de estados a seguir mostra as instruções expandidas.

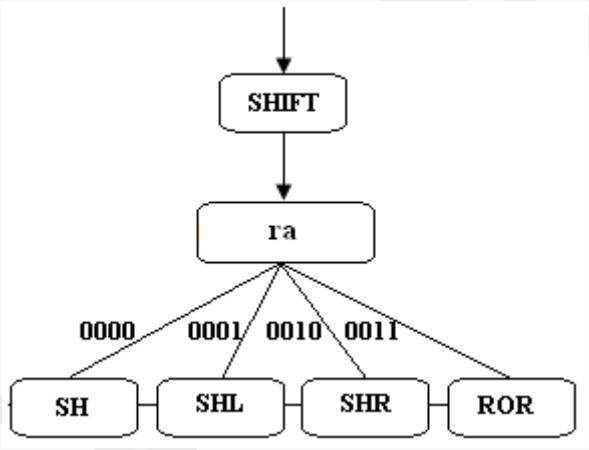


Quadro das palavras de controle geradas pela u.c. nas instruções de saltos.

Na execução da instrução de salto tanto condicional como incondicional os bits da instrução $d_7...d_0$ são carregados no PC se satisfeita a condição ou se a instrução for incondicional.

Instrução	JZ	JNZ	JMEQ	JNEQ	JC	JNC	JD0	JND0	JD7	JND7	JMOR	JEQ	JNOR	JMP
ra	0	1	3	5	6	7	8	9	A	B	C	D	E	F
PC	d	d	d	d	D	d	d	d	d	d	d	d	d	d

A instrução de deslocamento "shift" também será expandida como a seguir:



Quadro de palavras de controle gerados pela u.c. na execução da instrução de deslocamento "shift".

Instrução	ra	D_addr	D_rd	D_wr	RF_s	RF_w_addr	RF_w_wr	RF_Rp_rd	RF_Rp_addr	RF_Rq_rd	RF_Rq_addr	ALU	SH	OE
SH	0	0	0	0	0	rb	1	1	rc	0	xxxx	000	00	0
SHL	1	0	0	0	0	rb	1	1	rc	0	xxxx	000	01	0
SHR	2	0	0	0	0	rb	1	1	rc	0	xxxx	001	10	0
ROR	3	0	0	0	0	rb	1	1	rc	0	xxxx	111	11	0

Formato da instrução – A instrução tem 16bits subdivididos de 4 em 4 para informação de fonte e destino. Quando se utiliza a memória de dados são necessários 8bits para o endereçamento e a união das fontes rb e rc forma o d.

Opcode Instrução	Destino ra	Fonte rb	Fonte rc
$d_{15}..d_0$	$d_{11}...d_8$	$d_7,,d_4$	$d_3..d_0$

d(8bits) = rb(4bits) e rc(4bits).

Descrição de cada instrução.

1. LD(ra),(d) - Carrega da memória de dados o conteúdo endereçado pelo parâmetro d e transfere para o registrador de arquivo endereçado pelo endereço dado por ra.

Instrução	Mnem	opcode	Operação
Carrega	LD(ra) ← D(d)	0000	RF[ra] ← D(d)

2. ST(d),(ra) - Carrega na memória de dados o conteúdo endereçado pelo parâmetro d e transfere do registrador de arquivo endereçado pelo endereço dado por ra.

Instrução	Mnem	opcode	Operação
Armazena	ST(d) ← (ra)	0001	D(d) ← RF[ra]

3. ADD(ra),(rb),(rc) – Os conteúdos no registrador de arquivo dado pelos registradores b e c são somados na ULA e o resultado deve ser armazenado no próprio registrador de arquivo no endereço dado pelo registrador ra.

Instrução	Mnem	opcode	Operação
Carrega	ADD(ra),(rb),(rc)	0010	RF[ra] ← RF[rb] + RF[rc]

4. MVI(ra), #C - Carrega imediatamente o conteúdo c no registrador de arquivo no endereço fornecido pelo registrador ra.

Instrução	Mnem	opcode	Operação
Mover dado	MVI(ra) ← c	0011	RF[ra] ← c

5. SUB(ra),(rb),(rc) - Os conteúdos no registrador de arquivo dado pelos registradores b e c são subtraídos na ULA e o resultado deve ser armazenado no próprio registrador de arquivo no endereço dado pelo registrador ra.

Instrução	Mnem	opcode	Operação
Subtrai	SUB(ra),(rb),(rc)	0100	RF[ra] ← RF[Rb] – RF[rc]

6. JZ,(d) – Salta para o endereço dado em d se o resultado na saída da ULA for igual a zero.

Instrução	Mnem	opcode	Ra	Operação
Salto se zero	JZ,(d)	0101	0000	PC ← (d)

7. JNZ,(d) - Salta para o endereço dado em d se o resultado na saída da ULA for diferente de zero.

Instrução	Mnem	opcode	ra	Operação
Salto se não zero	JNZ,(d)	0101	0001	PC ← (d)

8. JNEQ,(d) - Salta para o endereço dado em d se a comparação entre conteúdos A e B for menor ou igual.

Instrução	Mnem	opcode	ra	Operação
Salto menor ou igual	JNEQ,(d)	0101	0011	PC ← (d)

9. JNEQ,(d) - Salta para o endereço dado em d se a comparação entre conteúdos A e B for maior ou igual.

Instrução	Mnem	opcode	ra	Operação
Salto maior ou igual	JMEQ,(d)	0101	0101	PC ← (d)

10. JC,(d) - Salta para o endereço dado em d se a a operação realizada na ULA gerou um bit vai um "carry" .

Instrução	Mnem	opcode	Ra	Operação
Salto se carry	JC,(d)	0101	0110	PC ← (d)

11. JNC,(d) - Salta para o endereço dado em d se a operação realizada na ULA não gerou um bit vai um "carry".

Instrução	Mnem	Opcode	ra	Operação
Salto se não carry	JNC,(d)	0101	0111	PC ← (d)

12. JD0,(d) - Salta para o endereço dado em d se o bit D0 do conteúdo no registrador de arquivo dado pelo registrador ra e deslocado para o deslocador é igual a "1'.

Instrução	Mnem	opcode	ra	Operação
Salto se bit D0	JD0(ra) ← (d)	0101	1000	PC ← (d)

13. JND0,(d) - Salta para o endereço dado em d se o bit D0 do conteúdo no registrador de arquivo dado pelo registrador ra e deslocado para o deslocador é igual a "0'.

Instrução	Mnem	opcode	ra	Operação
Salto se não bit D0	JND0(ra) ← (d)	0101	1001	PC ← (d)

14. JD7,(d) - Salta para o endereço dado em d se o bit D7 do conteúdo no registrador de arquivo dado pelo registrador ra e deslocado para o deslocador é igual a "1'.

Instrução	Mnem	opcode	ra	Operação
Salto se bit D7	JD7(ra) ← (d)	0101	1010	PC ← (d)

15. JND7,(d) - Salta para o endereço dado em d se o bit D0 do conteúdo no registrador de arquivo dado pelo registrador ra e deslocado para o deslocador é igual a "0'.

Instrução	Mnem	opcode	ra	Operação
Salto se não bit D7	JND7(ra) ← (d)	0101	1011	PC ← (d)

16. JMOR,(d) - Salta para o endereço dado em d se a comparação entre os conteúdos nos registradores de arquivo dados pelos endereços dos registradores ra e rb, for maiorl.

Instrução	Mnem	Opcode	ra	Operação
Salta se maior	JMOR(ra) ← (d)	0101	1100	PC ← (d)

17. JEQ,(d) - Salta para o endereço dado em d se a comparação entre os conteúdos nos registradores de arquivo dados pelos endereços dos registradores ra e rb, for igual.

Instrução	Mnem	Opcode	ra	Operação
Salto se igual	JEQ(ra) ← (d)	0101	1101	PC ← (d)

18. JNOR,(d) - Salta para o endereço dado em d se a comparação entre os conteúdos nos registradores de arquivo dados pelos endereços dos registradores ra e rb, for menor.

Instrução	Mnem	opcode	ra	Operação
Salto se menor	JNOR(ra) ← (d)	0101	1110	PC ← (d)

19. JMP,(d) - Salta para o endereço dado em d.

Instrução	Mnem	opcode	ra	Operação
Salto incondicional	JMP(ra) ← (d)	0101	1111	PC ← (d)

20. OUT(ra) – Transfere para a saída de dados o conteúdo do registrador de arquivo endereçado pelo registrador ra.

Instrução	Mnem	opcode	Operação
Saída	OUT(ra)	0110	OUT ← [ra]

21. IN(ra),ext - Carrega no registrador de arquivo no endereço dado pelo registrador ra o conteúdo externo.

Instrução	Mnem	opcode	Operação
Entrada	IN(ra) ← ext	0111	RF[ra] ← ext

22. CMP(ra),(rb) – Compara os conteúdos endereçados pelos registradores ra e rb no registrador de arquivo.

Instrução	Mnem	Opcode	Operação
Compare	CMP(ra),(rb)	1000	[ra] : [rb]

23. XOR(ra),(rb),(rc) - Os conteúdos no registrador de arquivo dado pelos registradores b e c são realizadas a operação exclusivo-ou na ULA e o resultado deve ser armazenado no próprio registrador de arquivo no endereço dado pelo registrador ra.

Instrução	Mnem	opcode	Operação
XOR	XOR(ra),(rb),(rc)	1001	RF[ra] ← RF[rb] ⊕ RF[rc]

24. AND(ra),(rb),(rc) - Os conteúdos no registrador de arquivo dado pelos registradores b e c são realizadas a operação "E" na ULA e o resultado deve ser armazenado no próprio registrador de arquivo no endereço dado pelo registrador ra.

Instrução	Mnem	opcode	Operação
E	AND(ra),(rb),(rc)	1010	RF[ra] ← RF[rb] ∧ RF[rc]

25. OR(ra),(rb),(rc) - Os conteúdos no registrador de arquivo dado pelos registradores b e c são realizadas a operação "OU" na ULA e o resultado deve ser armazenado no próprio registrador de arquivo no endereço dado pelo registrador ra.

Instrução	Mnem	Opcode	Operação
OU	OU(ra),(rb),(rc)	1011	RF[ra] ← RF[rb] ∨ RF[rc]

26. INC(ra) – Incrementa o conteúdo endereçado do registrador de arquivo endereçado pelo endereço dado por ra.

Instrução	Mnem	opcode	Operação
Incrementa	INC(ra)	1100	RF[ra] ← RF[ra] + 1

27. DEC(ra) - Decrementa o conteúdo endereçado do registrador de arquivo endereçado pelo endereço dado por ra.

Instrução	Mnem	opcode	Operação
Decrementa	DEC(ra)	1101	RF[ra] ← RF[ra] - 1

28. SH(rb),(rc) - Transfere o conteúdo do registrador de arquivo fornecido pelo registrador Rb e escreve o conteúdo no endereço fornecido pelo registrador rc no registrador de arquivo.

Instrução	Mnem	opcode	ra	Operação
Transfere	SH(ra),(rb)	1110	0000	RF[ra] ← (rc)

29. SHL(ra) – Desloca um bit a esquerda o conteúdo endereçado pelo registrador ra e preenche o bit com zero e armazena o conteúdo no endereço dado por ra.

Instrução	Mnem	opcode	ra	Operação
Desloca esquerda	SHL(ra) ← (d)	1110	0001	RF[ra] ← (d)

30. SHR(ra) - Desloca um bit a direita o conteúdo endereçado pelo registrador ra e preenche o bit com zero e armazena o conteúdo no endereço dado por ra.

Instrução	Mnem	opcode	ra	Operação
Desloca direita	SHR(ra) ← (d)	1110	0010	RF[ra] ← (d)

31. ROR(ra) – Gira um bit a direita o conteúdo endereçado pelo registrador ra e preenche o bit com bit deslocado e armazena o conteúdo no endereço dado por ra.

Instrução	Mnem	opcode	ra	Operação
Gira a direita	ROR(ra) ← (d)	1110	0011	RF[ra] ← (d7)

32. HLT - Carrega da memória de dados o conteúdo endereçado pelo parâmetro d e transfere para o registrador de arquivo endereçado pelo endereço dado por ra.

Instrução	Mnem	opcode	Operação
Parada	HLT	1111	PC ← PC

O próximo passo após concluído o projeto é a simulação e para isso precisamos criar vários programas envolvendo todo o conjunto de instruções e situações de decisões. Usaremos o quartus II para essa finalidade.